1장, 서울시 구별 CCTV현황 분석

1. CCTV와 인구현황 데이터 구하기
2. 파이썬 텍스트 파일과 엑셀 파일 읽어 오기(pandas)
3. CCTV와 인구 데이터 관리하기 (dataframe)
4. 데이터 합하기
5. 파이썬 시각화 도구 사용하기 (matplotlib)
6. 현황 그래프 표현하기

컴퓨터공학과 20154215 구명회

1. CCTV와 인구현황 데이터 구하기

In [1]:
```
import pandas as mh
```

In [2]:
```
CCTV_Seoul = mh.read_csv('../data/01. CCTV_in_Seoul.csv', encoding='utf-8')
CCTV_Seoul.head()
```

Out[2]:

	기관명	소계	2013년도 이전	2014년	2015년	2016년
0	강남구	2780	1292	430	584	932
1	강동구	773	379	99	155	377
2	강북구	748	369	120	138	204
3	강서구	884	388	258	184	81
4	관악구	1496	846	260	390	613

In [3]:
```
CCTV_Seoul.columns
```

Out[3]:
```
Index(['기관명', '소계', '2013년도 이전', '2014년', '2015년', '2016년'], dtype='object')
```

In [4]:
```
CCTV_Seoul.columns[0]
```

Out[4]:
```
'기관명'
```

In [5]:
```
CCTV_Seoul.rename(columns={CCTV_Seoul.columns[0] : '구별'}, inplace=True)
CCTV_Seoul.head()
```

Out[5]:

	구별	소계	2013년도 이전	2014년	2015년	2016년
0	강남구	2780	1292	430	584	932
1	강동구	773	379	99	155	377
2	강북구	748	369	120	138	204
3	강서구	884	388	258	184	81
4	관악구	1496	846	260	390	613

In [6]:
```
pop_Seoul = mh.read_excel('../data/01. population_in_Seoul.xls',  encoding='utf-8')
pop_Seoul.head()
```

Out[6]:

	기간	자치구	세대	인구	인구.1	인구.2	인구.3	인구.4	인구.5	인구.6	인구.7	인구.8	세대당인구	65세이상고령자
0	기간	자치구	세대	합계	합계	합계	한국인	한국인	한국인	등록외국인	등록외국인	등록외국인	세대당인구	65세이상고령자
1	기간	자치구	세대	계	남자	여자	계	남자	여자	계	남자	여자	세대당인구	65세이상고령자
2	2017.1/4	합계	4202888	10197604	5000005	5197599	9926968	4871560	5055408	270636	128445	142191	2.36	1321458
3	2017.1/4	종로구	72654	162820	79675	83145	153589	75611	77978	9231	4064	5167	2.11	25425
4	2017.1/4	중구	59481	133240	65790	67450	124312	61656	62656	8928	4134	4794	2.09	20764

In [7]:
```
pop_Seoul = mh.read_excel('../data/01. population_in_Seoul.xls', header = 2, usecols = 'B,D,G,J,N', encoding='utf-8')
pop_Seoul.head()
```

Out[7]:

	자치구	계	계.1	계.2	65세이상고령자
0	합계	10197604.0	9926968.0	270636.0	1321458.0
1	종로구	162820.0	153589.0	9231.0	25425.0
2	중구	133240.0	124312.0	8928.0	20764.0
3	용산구	244203.0	229456.0	14747.0	36231.0
4	성동구	311244.0	303380.0	7864.0	39997.0

In [8]:
```
pop_Seoul.rename(columns={pop_Seoul.columns[0] : '구별',
                          pop_Seoul.columns[1] : '인구수',
                          pop_Seoul.columns[2] : '한국인',
                          pop_Seoul.columns[3] : '외국인',
                          pop_Seoul.columns[4] : '고령자'}, inplace=True)
pop_Seoul.head()
```

Out[8]:

	구별	인구수	한국인	외국인	고령자
0	합계	10197604.0	9926968.0	270636.0	1321458.0
1	종로구	162820.0	153589.0	9231.0	25425.0
2	중구	133240.0	124312.0	8928.0	20764.0
3	용산구	244203.0	229456.0	14747.0	36231.0
4	성동구	311244.0	303380.0	7864.0	39997.0

2. 파이썬 텍스트 파일과 엑셀 파일 읽어 오기(pandas)

In [9]:
```
import pandas as mh
import numpy as mhnp
```

In [10]:
```
s = mh.Series([1,3,5,mhnp.nan, 6, 8])
s
```

Out[10]:
```
0    1.0
1    3.0
2    5.0
3    NaN
4    6.0
5    8.0
dtype: float64
```

In [11]:
```
dates = mh.date_range('20130101', periods=6)
dates
```

Out[11]:
```
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
               '2013-01-05', '2013-01-06'],
              dtype='datetime64[ns]', freq='D')
```

In [12]:
```
df = mh.DataFrame(mhnp.random.randn(6,4), index=dates, columns=['A','B','C','D'])
df
```

Out[12]:

	A	B	C	D
2013-01-01	1.055164	0.909075	0.411462	-0.650478
2013-01-02	0.585729	-0.333258	-1.286833	1.336482
2013-01-03	-0.335723	-0.661128	0.969986	-0.279669
2013-01-04	-1.094279	-0.752960	0.470274	-0.386912
2013-01-05	-0.469839	1.159787	1.324363	0.229972
2013-01-06	0.425568	0.375653	-0.370739	0.049519

In [13]:
```
df = mh.DataFrame(mhnp.random.randn(6,4), index=dates, columns=['A','B','C','D'])
df
```

Out[13]:

	A	B	C	D
2013-01-01	1.196405	-1.128347	0.558117	-0.347657
2013-01-02	-0.517543	-0.167052	-0.895728	0.410827
2013-01-03	1.834496	-0.671298	1.630622	0.048431
2013-01-04	-0.406784	0.220034	0.602620	0.518096
2013-01-05	-1.347131	0.597863	0.644538	-2.557825
2013-01-06	0.403527	1.455111	-2.035246	0.052808

In [14]:
```
df.head(3)
```

Out[14]:

	A	B	C	D
2013-01-01	1.196405	-1.128347	0.558117	-0.347657
2013-01-02	-0.517543	-0.167052	-0.895728	0.410827
2013-01-03	1.834496	-0.671298	1.630622	0.048431

In [15]:
```
df.head()
```

Out[15]:

	A	B	C	D
2013-01-01	1.196405	-1.128347	0.558117	-0.347657
2013-01-02	-0.517543	-0.167052	-0.895728	0.410827
2013-01-03	1.834496	-0.671298	1.630622	0.048431
2013-01-04	-0.406784	0.220034	0.602620	0.518096
2013-01-05	-1.347131	0.597863	0.644538	-2.557825

In [16]:
```
df.index
```

Out[16]:
```
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
               '2013-01-05', '2013-01-06'],
              dtype='datetime64[ns]', freq='D')
```

In [17]:
```
df.columns
```

Out[17]:
```
Index(['A', 'B', 'C', 'D'], dtype='object')
```

In [18]:
```
df.values
```

Out[18]:
```
array([[ 1.19640513, -1.12834716,  0.55811732, -0.34765676],
       [-0.51754312, -0.16705211, -0.89572806,  0.41082655],
       [ 1.83449586, -0.67129772,  1.63062218,  0.04843109],
       [-0.4067842 ,  0.22003355,  0.60261964,  0.51809554],
       [-1.34713118,  0.59786344,  0.64453834, -2.55782457],
       [ 0.40352685,  1.45511087, -2.0352459 ,  0.05280757]])
```

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 6 entries, 2013-01-01 to 2013-01-06
Freq: D
Data columns (total 4 columns):
A    6 non-null float64
B    6 non-null float64
C    6 non-null float64
D    6 non-null float64
dtypes: float64(4)
memory usage: 240.0 bytes
```

In [20]:
```
df.describe()
```

Out [20]:

	A	B	C	D
count	6.000000	6.000000	6.000000	6.000000
mean	0.193828	0.051052	0.084154	-0.312553
std	1.181716	0.922960	1.315376	1.141887
min	-1.347131	-1.128347	-2.035246	-2.557825
25%	-0.489853	-0.545236	-0.532267	-0.248635
50%	-0.001629	0.026491	0.580368	0.050619
75%	0.998186	0.503406	0.634059	0.321322
max	1.834496	1.455111	1.630622	0.518096

In [21]:
```
df.sort_values(by='B', ascending=False)
```

Out [21]:

	A	B	C	D
2013-01-06	0.403527	1.455111	-2.035246	0.052808
2013-01-05	-1.347131	0.597863	0.644538	-2.557825
2013-01-04	-0.406784	0.220034	0.602620	0.518096
2013-01-02	-0.517543	-0.167052	-0.895728	0.410827
2013-01-03	1.834496	-0.671298	1.630622	0.048431
2013-01-01	1.196405	-1.128347	0.558117	-0.347657

In [22]:
```
df.sort_values(by='A', ascending=True)
```

Out [22]:

	A	B	C	D
2013-01-05	-1.347131	0.597863	0.644538	-2.557825
2013-01-02	-0.517543	-0.167052	-0.895728	0.410827
2013-01-04	-0.406784	0.220034	0.602620	0.518096
2013-01-06	0.403527	1.455111	-2.035246	0.052808
2013-01-01	1.196405	-1.128347	0.558117	-0.347657
2013-01-03	1.834496	-0.671298	1.630622	0.048431

In [23]:
```
df #주석테스트
```

Out [23]:

	A	B	C	D
2013-01-01	1.196405	-1.128347	0.558117	-0.347657
2013-01-02	-0.517543	-0.167052	-0.895728	0.410827
2013-01-03	1.834496	-0.671298	1.630622	0.048431
2013-01-04	-0.406784	0.220034	0.602620	0.518096
2013-01-05	-1.347131	0.597863	0.644538	-2.557825
2013-01-06	0.403527	1.455111	-2.035246	0.052808

In [24]:
```
df['A']
```

Out [24]:
```
2013-01-01    1.196405
2013-01-02   -0.517543
2013-01-03    1.834496
2013-01-04   -0.406784
2013-01-05   -1.347131
2013-01-06    0.403527
Freq: D, Name: A, dtype: float64
```

In [25]:
```
df[0:3]
```

Out [25]:

	A	B	C	D
2013-01-01	1.196405	-1.128347	0.558117	-0.347657
2013-01-02	-0.517543	-0.167052	-0.895728	0.410827
2013-01-03	1.834496	-0.671298	1.630622	0.048431

In [26]:
```
df[1:5]
```

Out [26]:

	A	B	C	D
2013-01-02	-0.517543	-0.167052	-0.895728	0.410827
2013-01-03	1.834496	-0.671298	1.630622	0.048431
2013-01-04	-0.406784	0.220034	0.602620	0.518096
2013-01-05	-1.347131	0.597863	0.644538	-2.557825

In [27]:
```
df['20130102':'20130104']
```

Out [27]:

	A	B	C	D
2013-01-02	-0.517543	-0.167052	-0.895728	0.410827
2013-01-03	1.834496	-0.671298	1.630622	0.048431
2013-01-04	-0.406784	0.220034	0.602620	0.518096

In [28]:
```
df['20130101':'20130105']
```

Out [28]:

	A	B	C	D
2013-01-01	1.196405	-1.128347	0.558117	-0.347657
2013-01-02	-0.517543	-0.167052	-0.895728	0.410827
2013-01-03	1.834496	-0.671298	1.630622	0.048431
2013-01-04	-0.406784	0.220034	0.602620	0.518096
2013-01-05	-1.347131	0.597863	0.644538	-2.557825

In [29]:
```
df.loc[dates[0]]
```

Out[29]:
```
A    1.196405
B   -1.128347
C    0.558117
D   -0.347657
Name: 2013-01-01 00:00:00, dtype: float64
```

In [30]:
```
df.loc[:,['A','B']]
```

Out[30]:

	A	B
2013-01-01	1.196405	-1.128347
2013-01-02	-0.517543	-0.167052
2013-01-03	1.834496	-0.671298
2013-01-04	-0.406784	0.220034
2013-01-05	-1.347131	0.597863
2013-01-06	0.403527	1.455111

In [31]:
```
df.loc['20130102':'20130104',['A','B']]
```

Out[31]:

	A	B
2013-01-02	-0.517543	-0.167052
2013-01-03	1.834496	-0.671298
2013-01-04	-0.406784	0.220034

In [32]:
```
df.loc['20130102',['A','B']]
```

Out[32]:
```
A   -0.517543
B   -0.167052
Name: 2013-01-02 00:00:00, dtype: float64
```

In [33]:
```
df.loc[dates[0],'A']
```

Out[33]:
```
1.1964051294298712
```

In [34]:
```
df.iloc[3]
```

Out [34]:
```
A   -0.406784
B    0.220034
C    0.602620
D    0.518096
Name: 2013-01-04 00:00:00, dtype: float64
```

In [35]:
```
df.iloc[3:5, 0:2]
```

Out [35]:

	A	B
2013-01-04	-0.406784	0.220034
2013-01-05	-1.347131	0.597863

In [36]:
```
df.iloc[[1,2,4],[0,2]]
```

Out [36]:

	A	C
2013-01-02	-0.517543	-0.895728
2013-01-03	1.834496	1.630622
2013-01-05	-1.347131	0.644538

In [37]:
```
df.iloc[1:3,:]
```

Out [37]:

	A	B	C	D
2013-01-02	-0.517543	-0.167052	-0.895728	0.410827
2013-01-03	1.834496	-0.671298	1.630622	0.048431

In [38]:
```
df.iloc[:,1:3]
```

Out [38]:

	B	C
2013-01-01	-1.128347	0.558117
2013-01-02	-0.167052	-0.895728
2013-01-03	-0.671298	1.630622
2013-01-04	0.220034	0.602620
2013-01-05	0.597863	0.644538
2013-01-06	1.455111	-2.035246

In [39]:
```
df
```

Out[39]:

	A	B	C	D
2013-01-01	1.196405	-1.128347	0.558117	-0.347657
2013-01-02	-0.517543	-0.167052	-0.895728	0.410827
2013-01-03	1.834496	-0.671298	1.630622	0.048431
2013-01-04	-0.406784	0.220034	0.602620	0.518096
2013-01-05	-1.347131	0.597863	0.644538	-2.557825
2013-01-06	0.403527	1.455111	-2.035246	0.052808

In [40]:
```
df[df.A > 0]
```

Out[40]:

	A	B	C	D
2013-01-01	1.196405	-1.128347	0.558117	-0.347657
2013-01-03	1.834496	-0.671298	1.630622	0.048431
2013-01-06	0.403527	1.455111	-2.035246	0.052808

In [41]:
```
df[df>0]
```

Out[41]:

	A	B	C	D
2013-01-01	1.196405	NaN	0.558117	NaN
2013-01-02	NaN	NaN	NaN	0.410827
2013-01-03	1.834496	NaN	1.630622	0.048431
2013-01-04	NaN	0.220034	0.602620	0.518096
2013-01-05	NaN	0.597863	0.644538	NaN
2013-01-06	0.403527	1.455111	NaN	0.052808

In [42]:
```
df2 = df.copy()
```

In [43]:
```
df2['E'] = ['one', 'one', 'two', 'three', 'four', 'three']
df2
```

Out [43]:

	A	B	C	D	E
2013-01-01	1.196405	-1.128347	0.558117	-0.347657	one
2013-01-02	-0.517543	-0.167052	-0.895728	0.410827	one
2013-01-03	1.834496	-0.671298	1.630622	0.048431	two
2013-01-04	-0.406784	0.220034	0.602620	0.518096	three
2013-01-05	-1.347131	0.597863	0.644538	-2.557825	four
2013-01-06	0.403527	1.455111	-2.035246	0.052808	three

In [44]:
```
df2['E'].isin(['two','four'])
```

Out[44]:
```
2013-01-01    False
2013-01-02    False
2013-01-03     True
2013-01-04    False
2013-01-05     True
2013-01-06    False
Freq: D, Name: E, dtype: bool
```

In [45]:
```
df2['E'].isin(['one','three'])
```

Out[45]:
```
2013-01-01     True
2013-01-02     True
2013-01-03    False
2013-01-04     True
2013-01-05    False
2013-01-06     True
Freq: D, Name: E, dtype: bool
```

In [46]:
```
df2[df2['E'].isin(['two', 'four'])]
```

Out[46]:

	A	B	C	D	E
2013-01-03	1.834496	-0.671298	1.630622	0.048431	two
2013-01-05	-1.347131	0.597863	0.644538	-2.557825	four

In [47]:
```
df
```

Out[47]:

	A	B	C	D
2013-01-01	1.196405	-1.128347	0.558117	-0.347657
2013-01-02	-0.517543	-0.167052	-0.895728	0.410827
2013-01-03	1.834496	-0.671298	1.630622	0.048431
2013-01-04	-0.406784	0.220034	0.602620	0.518096
2013-01-05	-1.347131	0.597863	0.644538	-2.557825
2013-01-06	0.403527	1.455111	-2.035246	0.052808

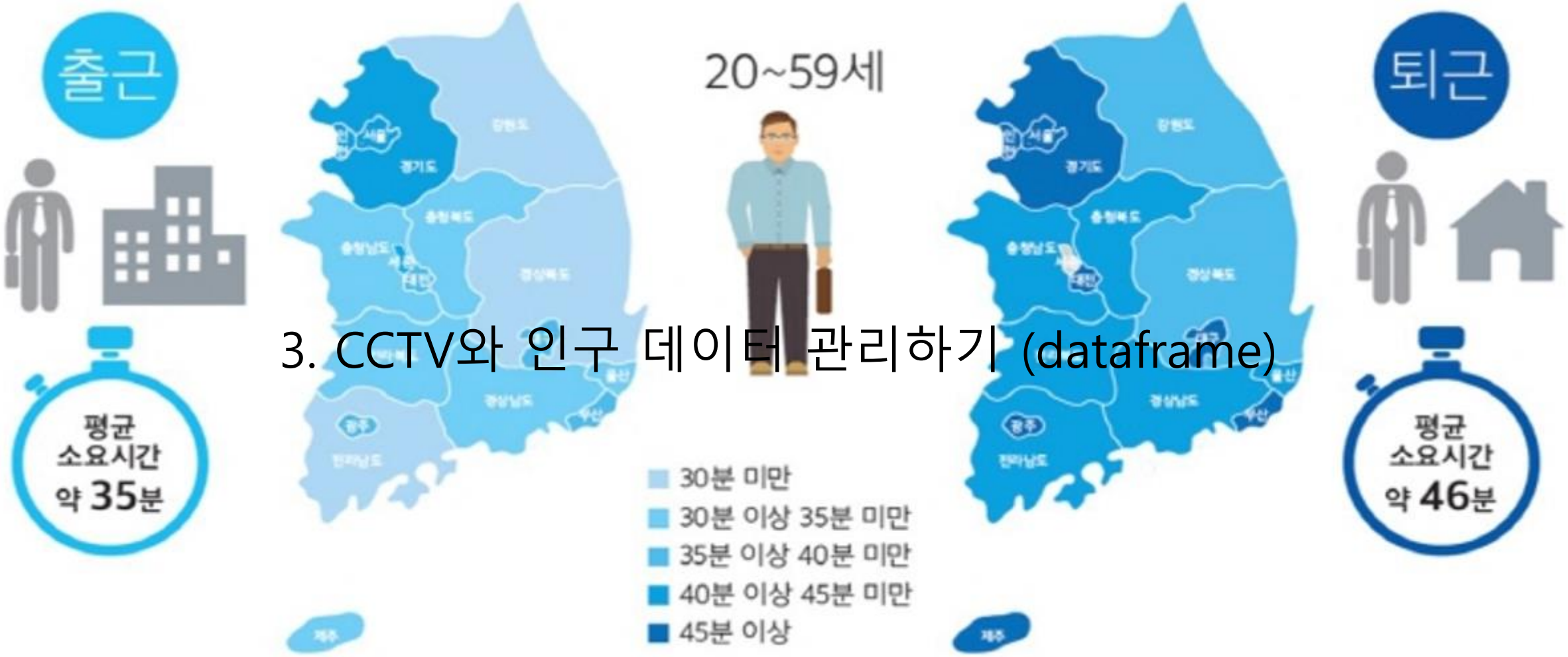
In [48]:
```
df.apply(mhnp.cumsum)
```

Out[48]:

	A	B	C	D
2013-01-01	1.196405	-1.128347	0.558117	-0.347657
2013-01-02	0.678862	-1.295399	-0.337611	0.063170
2013-01-03	2.513358	-1.966697	1.293011	0.111601
2013-01-04	2.106574	-1.746663	1.895631	0.629696
2013-01-05	0.759442	-1.148800	2.540169	-1.928128
2013-01-06	1.162969	0.306311	0.504924	-1.875321

In [49]:
```
df.apply(lambda x: x.max() - x.min())
```

Out[49]:
```
A    3.181627
B    2.583458
C    3.665868
D    3.075920
dtype: float64
```



3. CCTV와 인구 데이터 관리하기 (dataframe)

In [50]:
```
CCTV_Seoul.head()
```

Out [50]:

	구별	소계	2013년도 이전	2014년	2015년	2016년
0	강남구	2780	1292	430	584	932
1	강동구	773	379	99	155	377
2	강북구	748	369	120	138	204
3	강서구	884	388	258	184	81
4	관악구	1496	846	260	390	613

In [51]:
```
CCTV_Seoul.sort_values(by='소계', ascending=True).head()
```

Out [51]:

	구별	소계	2013년도 이전	2014년	2015년	2016년
9	도봉구	485	238	159	42	386
12	마포구	574	314	118	169	379
17	송파구	618	529	21	68	463
24	중랑구	660	509	121	177	109
23	중구	671	413	190	72	348

In [52]:
```
CCTV_Seoul.sort_values(by='소계', ascending=False).head()
```

Out [52]:

	구별	소계	2013년도 이전	2014년	2015년	2016년
0	강남구	2780	1292	430	584	932
18	양천구	2034	1843	142	30	467
14	서초구	1930	1406	157	336	398
21	은평구	1873	1138	224	278	468
20	용산구	1624	1368	218	112	398

In [53]:
```
CCTV_Seoul['최근증가율'] = (CCTV_Seoul['2016년'] + CCTV_Seoul['2015년'] +
                        CCTV_Seoul['2014년']) / CCTV_Seoul['2013년도 이전'] * 100
CCTV_Seoul.sort_values(by='최근증가율', ascending=False).head()
```

Out [53]:

	구별	소계	2013년도 이전	2014년	2015년	2016년	최근증가율
22	종로구	1002	464	314	211	630	248.922414
9	도봉구	485	238	159	42	386	246.638655
12	마포구	574	314	118	169	379	212.101911
8	노원구	1265	542	57	451	516	188.929889
1	강동구	773	379	99	155	377	166.490765

In [54]:
```
pop_Seoul.head()
```

Out [54]:

	구별	인구수	한국인	외국인	고령자
0	합계	10197604.0	9926968.0	270636.0	1321458.0
1	종로구	162820.0	153589.0	9231.0	25425.0
2	중구	133240.0	124312.0	8928.0	20764.0
3	용산구	244203.0	229456.0	14747.0	36231.0
4	성동구	311244.0	303380.0	7864.0	39997.0

In [55]:
```
pop_Seoul.drop([0], inplace=True)
pop_Seoul.head()
```

Out [55]:

	구별	인구수	한국인	외국인	고령자
1	종로구	162820.0	153589.0	9231.0	25425.0
2	중구	133240.0	124312.0	8928.0	20764.0
3	용산구	244203.0	229456.0	14747.0	36231.0
4	성동구	311244.0	303380.0	7864.0	39997.0
5	광진구	372164.0	357211.0	14953.0	42214.0

In [56]:
```
pop_Seoul['구별'].unique()
```

Out [56]:
```
array(['종로구', '중구', '용산구', '성동구', '광진구', '동대문구', '중랑구', '성북구', '강북구',
       '도봉구', '노원구', '은평구', '서대문구', '마포구', '양천구', '강서구', '구로구', '금천구',
       '영등포구', '동작구', '관악구', '서초구', '강남구', '송파구', '강동구', nan],
      dtype=object)
```

In [57]:
```
pop_Seoul[pop_Seoul['구별'].isnull()]
```

Out [57]:

	구별	인구수	한국인	외국인	고령자
26	NaN	NaN	NaN	NaN	NaN

In [58]:
```
pop_Seoul.drop([26], inplace=True)
pop_Seoul.head()
```

Out [58]:

	구별	인구수	한국인	외국인	고령자
1	종로구	162820.0	153589.0	9231.0	25425.0
2	중구	133240.0	124312.0	8928.0	20764.0
3	용산구	244203.0	229456.0	14747.0	36231.0
4	성동구	311244.0	303380.0	7864.0	39997.0
5	광진구	372164.0	357211.0	14953.0	42214.0

In [59]:
```
pop_Seoul['외국인비율'] = pop_Seoul['외국인'] / pop_Seoul['인구수'] * 100
pop_Seoul['고령자비율'] = pop_Seoul['고령자'] / pop_Seoul['인구수'] * 100
pop_Seoul.head()
```

Out [59]:

	구별	인구수	한국인	외국인	고령자	외국인비율	고령자비율
1	종로구	162820.0	153589.0	9231.0	25425.0	5.669451	15.615404
2	중구	133240.0	124312.0	8928.0	20764.0	6.700690	15.583909
3	용산구	244203.0	229456.0	14747.0	36231.0	6.038828	14.836427
4	성동구	311244.0	303380.0	7864.0	39997.0	2.526635	12.850689
5	광진구	372164.0	357211.0	14953.0	42214.0	4.017852	11.342849

In [60]:
```
pop_Seoul.sort_values(by='인구수', ascending=False).head()
```

Out [60]:

	구별	인구수	한국인	외국인	고령자	외국인비율	고령자비율
24	송파구	667483.0	660584.0	6899.0	72506.0	1.033584	10.862599
16	강서구	603772.0	597248.0	6524.0	72548.0	1.080540	12.015794
23	강남구	570500.0	565550.0	4950.0	63167.0	0.867660	11.072217
11	노원구	569384.0	565565.0	3819.0	71941.0	0.670725	12.634883
21	관악구	525515.0	507203.0	18312.0	68082.0	3.484582	12.955291

In [61]:
```
pop_Seoul.sort_values(by='외국인', ascending=False).head()
```

Out [61]:

	구별	인구수	한국인	외국인	고령자	외국인비율	고령자비율
19	영등포구	402985.0	368072.0	34913.0	52413.0	8.663598	13.006191
17	구로구	447874.0	416487.0	31387.0	56833.0	7.007998	12.689506
18	금천구	255082.0	236353.0	18729.0	32970.0	7.342345	12.925255
21	관악구	525515.0	507203.0	18312.0	68082.0	3.484582	12.955291
6	동대문구	369496.0	354079.0	15417.0	54173.0	4.172440	14.661322

In [62]:
```
pop_Seoul.sort_values(by='외국인비율', ascending=False).head()
```

Out [62]:

	구별	인구수	한국인	외국인	고령자	외국인비율	고령자비율
19	영등포구	402985.0	368072.0	34913.0	52413.0	8.663598	13.006191
18	금천구	255082.0	236353.0	18729.0	32970.0	7.342345	12.925255
17	구로구	447874.0	416487.0	31387.0	56833.0	7.007998	12.689506
2	중구	133240.0	124312.0	8928.0	20764.0	6.700690	15.583909
3	용산구	244203.0	229456.0	14747.0	36231.0	6.038828	14.836427

In [63]:
```
pop_Seoul.sort_values(by='고령자', ascending=False).head()
```

Out [63]:

	구별	인구수	한국인	외국인	고령자	외국인비율	고령자비율
16	강서구	603772.0	597248.0	6524.0	72548.0	1.080540	12.015794
24	송파구	667483.0	660584.0	6899.0	72506.0	1.033584	10.862599
12	은평구	494388.0	489943.0	4445.0	72334.0	0.899091	14.631019
11	노원구	569384.0	565565.0	3819.0	71941.0	0.670725	12.634883
21	관악구	525515.0	507203.0	18312.0	68082.0	3.484582	12.955291

In [64]:
```
pop_Seoul.sort_values(by='고령자비율', ascending=False).head()
```

Out [64]:

	구별	인구수	한국인	외국인	고령자	외국인비율	고령자비율
9	강북구	330192.0	326686.0	3506.0	54813.0	1.061806	16.600342
1	종로구	162820.0	153589.0	9231.0	25425.0	5.669451	15.615404
2	중구	133240.0	124312.0	8928.0	20764.0	6.700690	15.583909
3	용산구	244203.0	229456.0	14747.0	36231.0	6.038828	14.836427
13	서대문구	327163.0	314982.0	12181.0	48161.0	3.723221	14.720797

4. 데이터 합하기

4-1. 연습용 DataFrame 병합
4-2. 데이터 병합

In [65]:
```
mhdf1 = mh.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                      'B': ['B0', 'B1', 'B2', 'B3'],
                      'C': ['C0', 'C1', 'C2', 'C3'],
                      'D': ['D0', 'D1', 'D2', 'D3']},
                     index=[0, 1, 2, 3])

mhdf2 = mh.DataFrame({'A': ['A4', 'A5', 'A6', 'A7'],
                      'B': ['B4', 'B5', 'B6', 'B7'],
                      'C': ['C4', 'C5', 'C6', 'C7'],
                      'D': ['D4', 'D5', 'D6', 'D7']},
                     index=[4, 5, 6, 7])

mhdf3 = mh.DataFrame({'A': ['A8', 'A9', 'A10', 'A11'],
                      'B': ['B8', 'B9', 'B10', 'B11'],
                      'C': ['C8', 'C9', 'C10', 'C11'],
                      'D': ['D8', 'D9', 'D10', 'D11']},
                     index=[8, 9, 10, 11])
```

In [66]:
```
mhdf1
```

Out [66]:

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

In [67]:
```
mhdf2
```

Out [67]:

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

In [68]:
```
mhdf3
```

Out [68]:

	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

In [69]:
```
result = mh.concat([mhdf1, mhdf2, mhdf3])
result
```

Out [69]:

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

In [70]:
```
result = mh.concat([mhdf1, mhdf2, mhdf3], keys=['first', 'second', 'third'])
result
```

Out [70]:

		A	B	C	D
first	**0**	A0	B0	C0	D0
	1	A1	B1	C1	D1
	2	A2	B2	C2	D2
	3	A3	B3	C3	D3
second	**4**	A4	B4	C4	D4
	5	A5	B5	C5	D5
	6	A6	B6	C6	D6
	7	A7	B7	C7	D7
third	**8**	A8	B8	C8	D8
	9	A9	B9	C9	D9
	10	A10	B10	C10	D10
	11	A11	B11	C11	D11

In [71]:
```
result.index
```

Out[71]:
```
MultiIndex(levels=[['first', 'second', 'third'], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]],
           codes=[[0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]])
```

In [72]:
```
result.index.get_level_values(0)
```

Out[72]:
```
Index(['first', 'first', 'first', 'first', 'second', 'second', 'second',
       'second', 'third', 'third', 'third', 'third'],
      dtype='object')
```

In [73]:
```
result.index.get_level_values(1)
```

Out[73]:
```
Int64Index([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11], dtype='int64')
```

In [74]:
```
mhdf4 =mh.DataFrame({'B': ['B2', 'B3', 'B6', 'B7'],
                     'D': ['D2', 'D3', 'D6', 'D7'],
                     'F': ['F2', 'F3', 'F6', 'F7']},
                    index=[2, 3, 6, 7])

result = mh.concat([mhdf1, mhdf4], axis=1)
```

In [75]:
```
mhdf1
```

Out [75]:

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

In [76]:
```
mhdf4
```

Out [76]:

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

In [77]:
```
result
```

Out [77]:

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3
6	NaN	NaN	NaN	NaN	B6	D6	F6
7	NaN	NaN	NaN	NaN	B7	D7	F7

In [78]:
```
result = mh.concat([mhdf1, mhdf4], axis=1, join='inner')
result
```

Out [78]:

	A	B	C	D	B	D	F
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3

In [79]:
```
result = mh.concat([mhdf1, mhdf4], axis=1, join_axes=[mhdf1.index])
result
```

Out [79]:

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3

In [80]:
```
result = mh.concat([mhdf1, mhdf4], ignore_index=True)
result
#https://stackoverflow.com/questions/50501787/python-pandas-user-warning-sorting-because-non-concatenation-axis-is-not-aligne
```

```
C:₩Users₩nerin₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:1: FutureWarning: Sorting because non-concatenation axis is not aligned.
A future version
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.

  """Entry point for launching an IPython kernel.
```

Out [80]:

	A	B	C	D	F
0	A0	B0	C0	D0	NaN
1	A1	B1	C1	D1	NaN
2	A2	B2	C2	D2	NaN
3	A3	B3	C3	D3	NaN
4	NaN	B2	NaN	D2	F2
5	NaN	B3	NaN	D3	F3
6	NaN	B6	NaN	D6	F6
7	NaN	B7	NaN	D7	F7

https://stackoverflow.com/questions/50501787/python-pandas-user-warning-sorting-because-non-concatenation-axis-is-not-aligne

In [81]:
```
result = mh.concat([mhdf1, mhdf4], ignore_index=True, sort=True)
result
```

Out[81]:

	A	B	C	D	F
0	A0	B0	C0	D0	NaN
1	A1	B1	C1	D1	NaN
2	A2	B2	C2	D2	NaN
3	A3	B3	C3	D3	NaN
4	NaN	B2	NaN	D2	F2
5	NaN	B3	NaN	D3	F3
6	NaN	B6	NaN	D6	F6
7	NaN	B7	NaN	D7	F7

In [82]:
```
left = mh.DataFrame({'key': ['K0', 'K4', 'K2', 'K3'],
                     'A': ['A0', 'A1', 'A2', 'A3'],
                     'B': ['B0', 'B1', 'B2', 'B3']})

right = mh.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                      'C': ['C0', 'C1', 'C2', 'C3'],
                      'D': ['D0', 'D1', 'D2', 'D3']})
```

In [83]:
```
left
```

Out[83]:

	key	A	B
0	K0	A0	B0
1	K4	A1	B1
2	K2	A2	B2
3	K3	A3	B3

In [84]:
```
right
```

Out[84]:

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

In [85]:
```
mh.merge(left, right, on='key')
```

Out[85]:

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K2	A2	B2	C2	D2
2	K3	A3	B3	C3	D3

In [86]:
```
mh.merge(left, right, how='left', on='key')
```

Out[86]:

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K4	A1	B1	NaN	NaN
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

In [87]:
```
mh.merge(left, right, how='right', on='key')
```

Out[87]:

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K2	A2	B2	C2	D2
2	K3	A3	B3	C3	D3
3	K1	NaN	NaN	C1	D1

In [88]:
```
mh.merge(left, right, how='outer', on='key')
```

Out[88]:

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K4	A1	B1	NaN	NaN
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3
4	K1	NaN	NaN	C1	D1

In [89]:
```
mh.merge(left, right, how='inner', on='key')
```

Out[89]:

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K2	A2	B2	C2	D2
2	K3	A3	B3	C3	D3

In [90]:
```
data_result = mh.merge(CCTV_Seoul, pop_Seoul, on='구별')
data_result.head()
```

Out[90]:

	구별	소계	2013년도 이전	2014년	2015년	2016년	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율
0	강남구	2780	1292	430	584	932	150.619195	570500.0	565550.0	4950.0	63167.0	0.867660	11.072217
1	강동구	773	379	99	155	377	166.490765	453233.0	449019.0	4214.0	54622.0	0.929765	12.051638
2	강북구	748	369	120	138	204	125.203252	330192.0	326686.0	3506.0	54813.0	1.061806	16.600342
3	강서구	884	388	258	184	81	134.793814	603772.0	597248.0	6524.0	72548.0	1.080540	12.015794
4	관악구	1496	846	260	390	613	149.290780	525515.0	507203.0	18312.0	68082.0	3.484582	12.955291

In [91]:
```
del data_result['2013년도 이전']
del data_result['2014년']
del data_result['2015년']
del data_result['2016년']
data_result.head()
```

Out[91]:

	구별	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율
0	강남구	2780	150.619195	570500.0	565550.0	4950.0	63167.0	0.867660	11.072217
1	강동구	773	166.490765	453233.0	449019.0	4214.0	54622.0	0.929765	12.051638
2	강북구	748	125.203252	330192.0	326686.0	3506.0	54813.0	1.061806	16.600342
3	강서구	884	134.793814	603772.0	597248.0	6524.0	72548.0	1.080540	12.015794
4	관악구	1496	149.290780	525515.0	507203.0	18312.0	68082.0	3.484582	12.955291

In [92]:
```
data_result.set_index('구별', inplace=True)
data_result.head()
```

Out[92]:

구별	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율
강남구	2780	150.619195	570500.0	565550.0	4950.0	63167.0	0.867660	11.072217
강동구	773	166.490765	453233.0	449019.0	4214.0	54622.0	0.929765	12.051638
강북구	748	125.203252	330192.0	326686.0	3506.0	54813.0	1.061806	16.600342
강서구	884	134.793814	603772.0	597248.0	6524.0	72548.0	1.080540	12.015794
관악구	1496	149.290780	525515.0	507203.0	18312.0	68082.0	3.484582	12.955291

In [93]:
```
mhnp.corrcoef(data_result['고령자비율'],data_result['소계'])
```

Out[93]:
```
array([[ 1.        , -0.28078554],
       [-0.28078554,  1.        ]])
```

In [94]:
```
mhnp.corrcoef(data_result['외국인비율'],data_result['소계'])
```

Out[94]:
```
array([[ 1.        , -0.13607433],
       [-0.13607433,  1.        ]])
```

In [95]:
```
mhnp.corrcoef(data_result['인구수'],data_result['소계'])
```

Out[95]:
```
array([[1.        , 0.30634228],
       [0.30634228, 1.        ]])
```

In [96]:
```
data_result.sort_values(by='소계', ascending=False).head()
```

Out[96]:

구별	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율
강남구	2780	150.619195	570500.0	565550.0	4950.0	63167.0	0.867660	11.072217
양천구	2034	34.671731	479978.0	475949.0	4029.0	52975.0	0.839413	11.036964
서초구	1930	63.371266	450310.0	445994.0	4316.0	51733.0	0.958451	11.488308
은평구	1873	85.237258	494388.0	489943.0	4445.0	72334.0	0.899091	14.631019
용산구	1624	53.216374	244203.0	229456.0	14747.0	36231.0	6.038828	14.836427

In [97]:
```
data_result.sort_values(by='인구수', ascending=False).head()
###
```

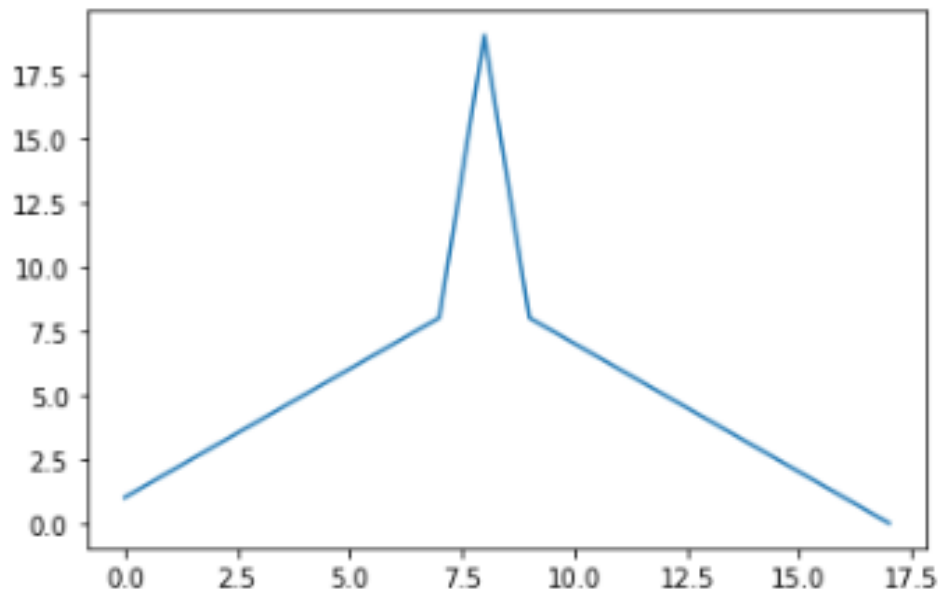
Out[97]:

구별	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율
송파구	618	104.347826	667483.0	660584.0	6899.0	72506.0	1.033584	10.862599
강서구	884	134.793814	603772.0	597248.0	6524.0	72548.0	1.080540	12.015794
강남구	2780	150.619195	570500.0	565550.0	4950.0	63167.0	0.867660	11.072217
노원구	1265	188.929889	569384.0	565565.0	3819.0	71941.0	0.670725	12.634883
관악구	1496	149.290780	525515.0	507203.0	18312.0	68082.0	3.484582	12.955291

5. 파이썬 시각화 도구 사용하기 (matplotlib)

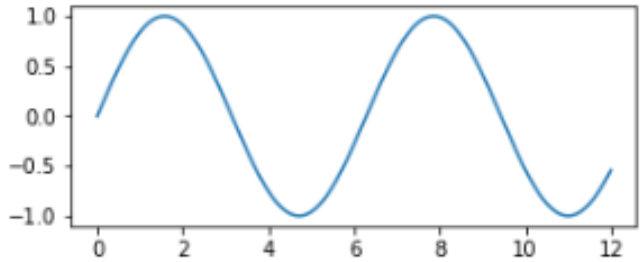
In [98]:
```
import matplotlib.pyplot as mhplt
%matplotlib inline
```

In [99]:
```
mhplt.figure()
mhplt.plot([1,2,3,4,5,6,7,8,19,8,7,6,5,4,3,2,1,0])
mhplt.show()
```

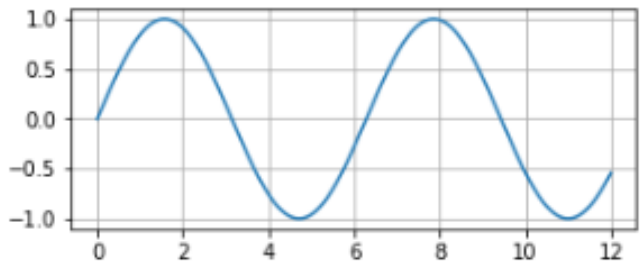


In [100]:
```
import numpy as mhnp

t = mhnp.arange(0,12,0.01)

y = mhnp.sin(t)
```

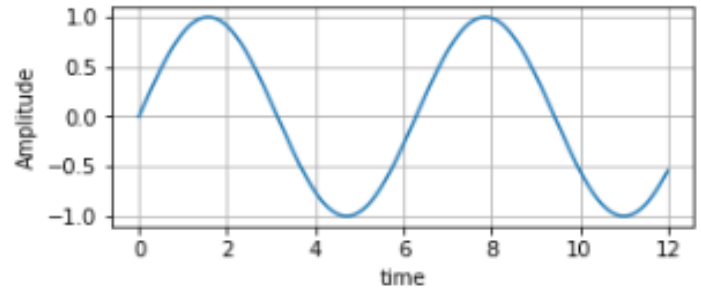
In [101]:
```
mhplt.figure(figsize=(5,2))
mhplt.plot(t, y)
mhplt.show()
```



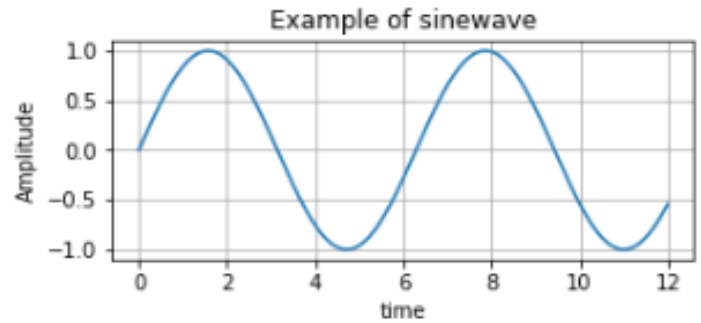
In [102]:
```
mhplt.figure(figsize=(5,2))
mhplt.plot(t, y)
mhplt.grid() # 그리드 적용하기
mhplt.show()
```



In [103]:
```
mhplt.figure(figsize=(5,2))
mhplt.plot(t, y)
mhplt.grid()
mhplt.xlabel('time')       # x축 라벨 적용하기
mhplt.ylabel('Amplitude')  # y축 라벨 적용하기
mhplt.show()
```



In [104]:
```
mhplt.figure(figsize=(5,2))
mhplt.plot(t, y)
mhplt.grid()
mhplt.xlabel('time')
mhplt.ylabel('Amplitude')
mhplt.title('Example of sinewave')
mhplt.show()
```

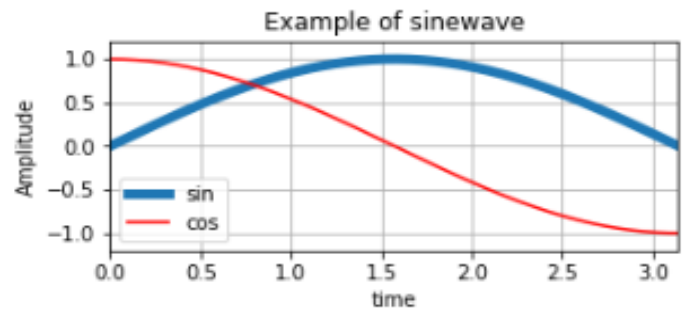


In [105]:
```
mhplt.figure(figsize=(5,2))
mhplt.plot(t, mhnp.sin(t))
mhplt.plot(t, mhnp.cos(t))
mhplt.grid()
mhplt.xlabel('time')
mhplt.ylabel('Amplitude')
mhplt.title('Example of sinewave')
mhplt.show()
```

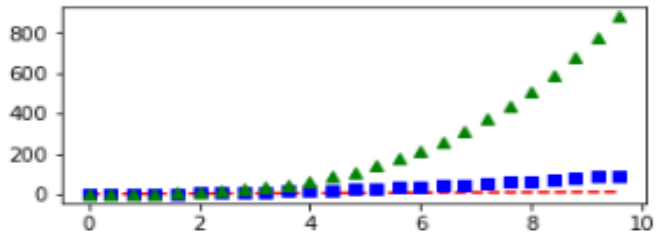
In [106]:
```
mhplt.figure(figsize=(5,2))
mhplt.plot(t, mhnp.sin(t), label='sin')
mhplt.plot(t, mhnp.cos(t), label='cos')
mhplt.grid()
mhplt.legend()
mhplt.xlabel('time')
mhplt.ylabel('Amplitude')
mhplt.title('Example of sinewave')
mhplt.show()
```

In [107]:
```
mhplt.figure(figsize=(5,2))
mhplt.plot(t, mhnp.sin(t), lw=5, label='sin')
mhplt.plot(t, mhnp.cos(t), 'r', label='cos')
mhplt.grid()
mhplt.legend()
mhplt.xlabel('time')
mhplt.ylabel('Amplitude')
mhplt.title('Example of sinewave')
mhplt.show()
```

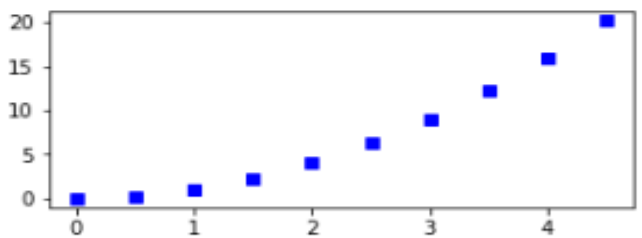
In [108]:
```
mhplt.figure(figsize=(5,2))
mhplt.plot(t, mhnp.sin(t), lw=5, label='sin')
mhplt.plot(t, mhnp.cos(t), 'r', label='cos')
mhplt.grid()
mhplt.legend()
mhplt.xlabel('time')
mhplt.ylabel('Amplitude')
mhplt.title('Example of sinewave')
mhplt.ylim(-1.2, 1.2)
mhplt.xlim(0, mhnp.pi)
mhplt.show()
```

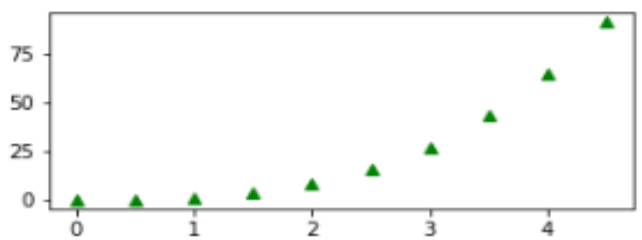


In [109]:
```
t = mhnp.arange(0, 10, 0.4)

mhplt.figure(figsize=(5,2))
mhplt.plot(t, t, 'r--')
mhplt.plot(t, t**2, 'bs')
mhplt.plot(t, t**3, 'g^')
mhplt.show()
```

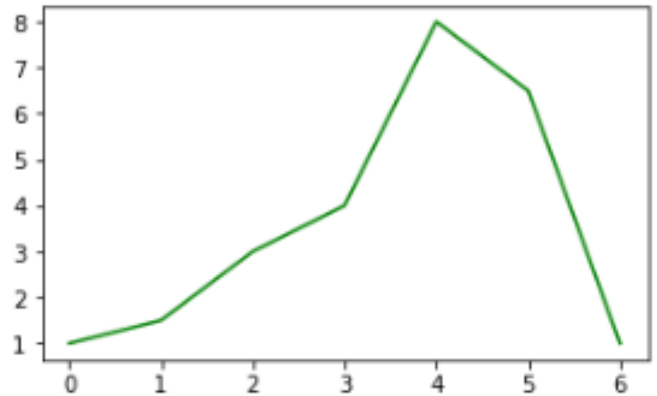


In [110]:
```
t = mhnp.arange(0, 5, 0.5)

mhplt.figure(figsize=(5,2))
pl1 = mhplt.plot(t, t**2, 'bs')

mhplt.figure(figsize=(5,2))
pl2 = mhplt.plot(t, t**3, 'g^')

mhplt.show()
```

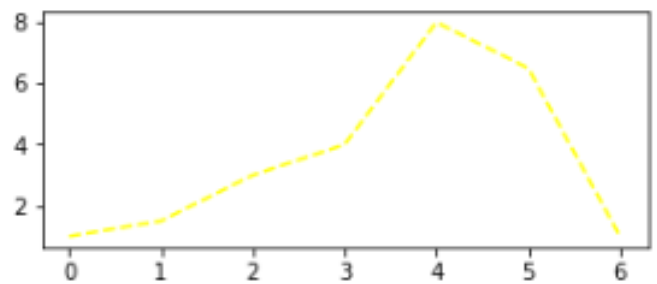




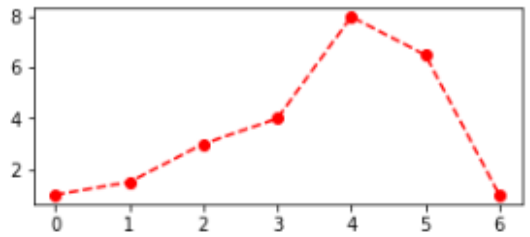
In [111]:
```
t = [0, 1, 2, 3, 4, 5, 6]
y = [1, 1.5, 3, 4, 8, 6.5, 1]

mhplt.figure(figsize=(5,3))
mhplt.plot(t, y, color='green')
mhplt.show()
```



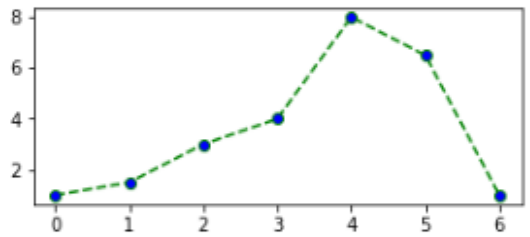
In [112]:
```
mhplt.figure(figsize=(5,2))
mhplt.plot(t, y, color='yellow', linestyle='dashed')
mhplt.show()
```



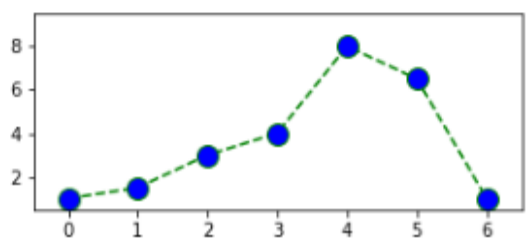
In [113]:
```
mhplt.figure(figsize=(5,2))
mhplt.plot(t, y, color='red', linestyle='dashed', marker='o')
mhplt.show()
```



In [114]:
```
mhplt.figure(figsize=(5,2))
mhplt.plot(t, y, color='green', linestyle='dashed', marker='o',
        markerfacecolor = 'blue')
mhplt.show()
```

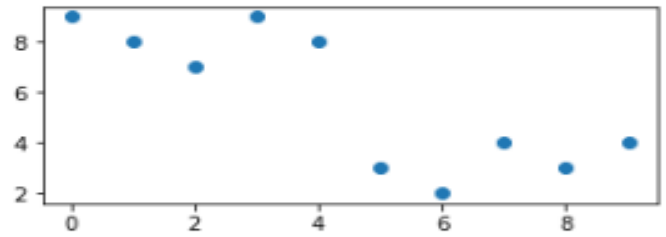


In [115]:
```
mhplt.figure(figsize=(5,2))
mhplt.plot(t, y, color='green', linestyle='dashed', marker='o',
        markerfacecolor = 'blue', markersize=12)

mhplt.xlim([-0.5, 6.5])
mhplt.ylim([0.5, 9.5])
mhplt.show()
```

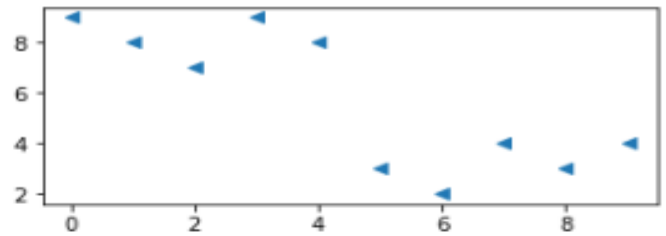


In [116]:
```
t = mhnp.array([0,1,2,3,4,5,6,7,8,9])
y = mhnp.array([9,8,7,9,8,3,2,4,3,4])
```

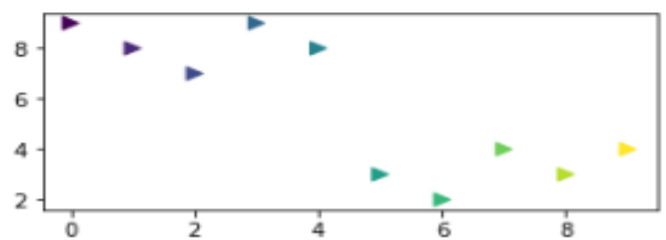
In [117]:
```
mhplt.figure(figsize=(5,2))
mhplt.scatter(t,y)
mhplt.show()
```



In [118]:
```
mhplt.figure(figsize=(5,2))
mhplt.scatter(t,y, marker='<')
mhplt.show()
```

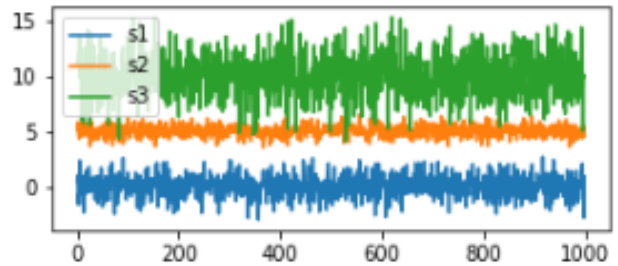


In [119]:
```
colormap = t

mhplt.figure(figsize=(5,2))
mhplt.scatter(t,y, s = 50, c = colormap, marker='>')
mhplt.show()
```

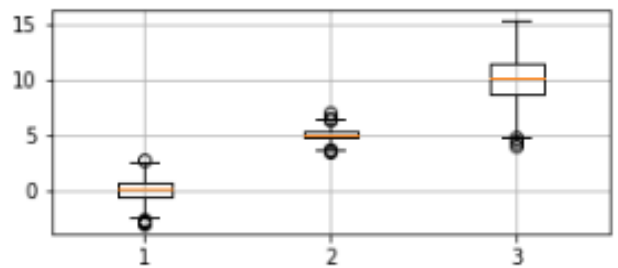


In [120]:
```
s1 = mhnp.random.normal(loc=0, scale=1, size=1000)
s2 = mhnp.random.normal(loc=5, scale=0.5, size=1000)
s3 = mhnp.random.normal(loc=10, scale=2, size=1000)
```

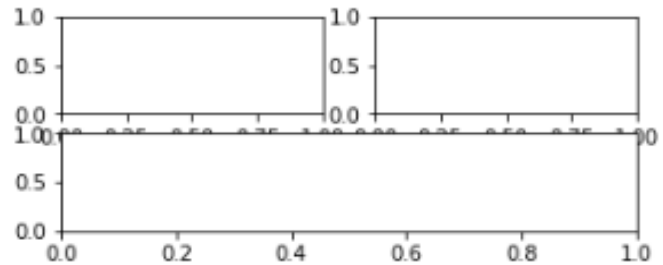
In [121]:
```
mhplt.figure(figsize=(5,2))
mhplt.plot(s1, label='s1')
mhplt.plot(s2, label='s2')
mhplt.plot(s3, label='s3')
mhplt.legend()
mhplt.show()
```



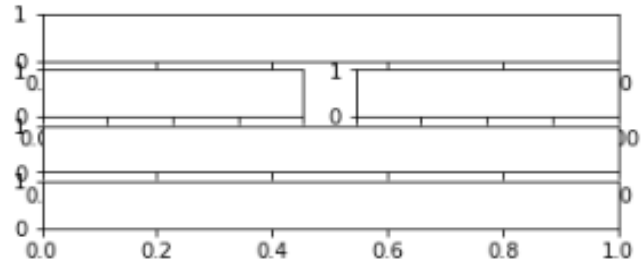
In [122]:
```
mhplt.figure(figsize=(5,2))
mhplt.boxplot((s1, s2, s3))
mhplt.grid()
mhplt.show()
```



In [123]:
```
mhplt.figure(figsize=(5,2))

mhplt.subplot(221)
mhplt.subplot(222)
mhplt.subplot(212)

mhplt.show()
```

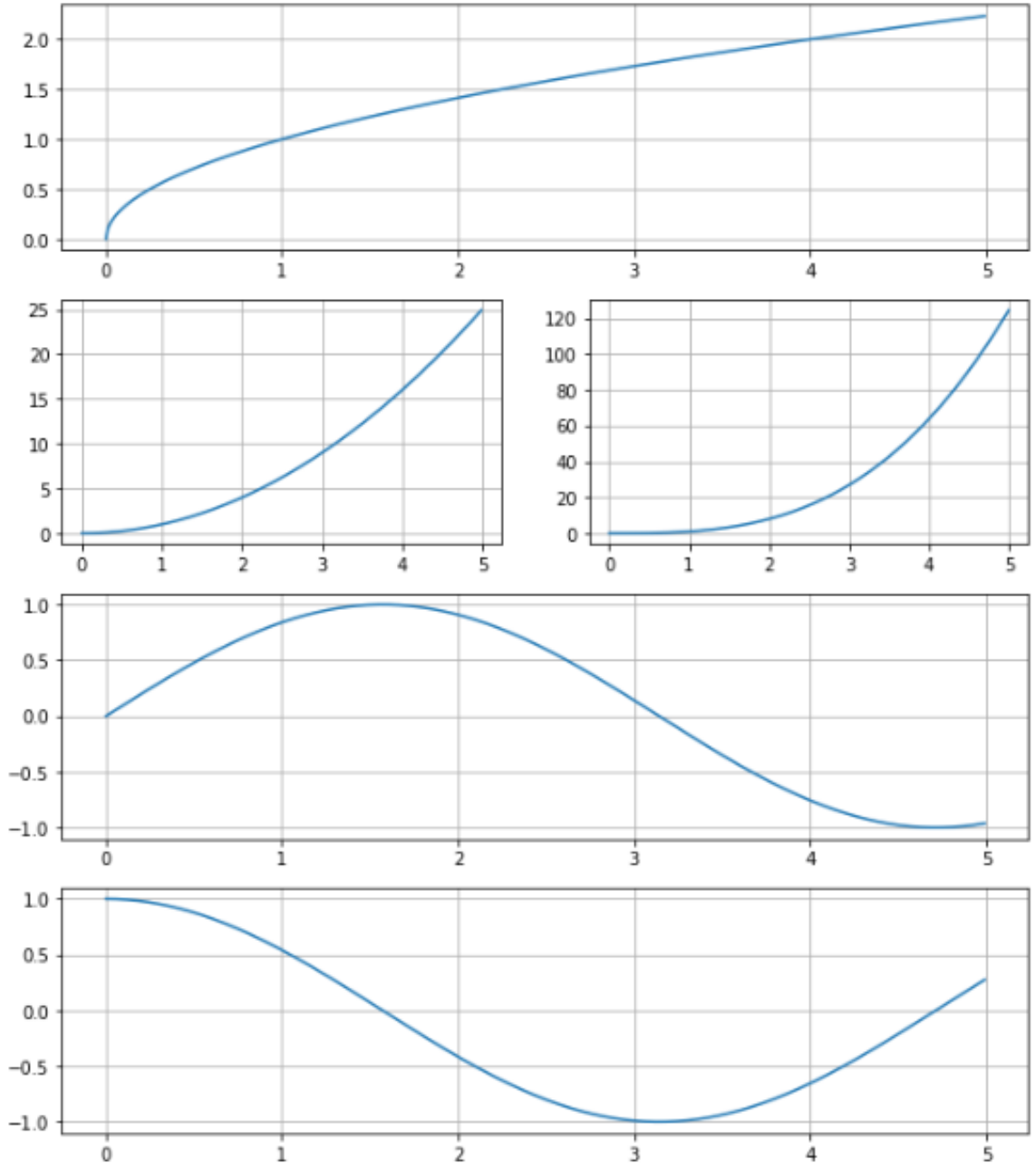


In [124]:
```
mhplt.figure(figsize=(5,2))

mhplt.subplot(411)
mhplt.subplot(423)
mhplt.subplot(424)
mhplt.subplot(413)
mhplt.subplot(414)

mhplt.show()
```



In [125]:

```
t = mhnp.arange(0,5,0.01)

mhplt.figure(figsize=(10,12))

mhplt.subplot(411)
mhplt.plot(t,mhnp.sqrt(t))
mhplt.grid()

mhplt.subplot(423)
mhplt.plot(t,t**2)
mhplt.grid()

mhplt.subplot(424)
mhplt.plot(t,t**3)
mhplt.grid()

mhplt.subplot(413)
mhplt.plot(t,mhnp.sin(t))
mhplt.grid()

mhplt.subplot(414)
mhplt.plot(t,mhnp.cos(t))
mhplt.grid()

mhplt.show()
```



6. 현황 그래프 표현하기

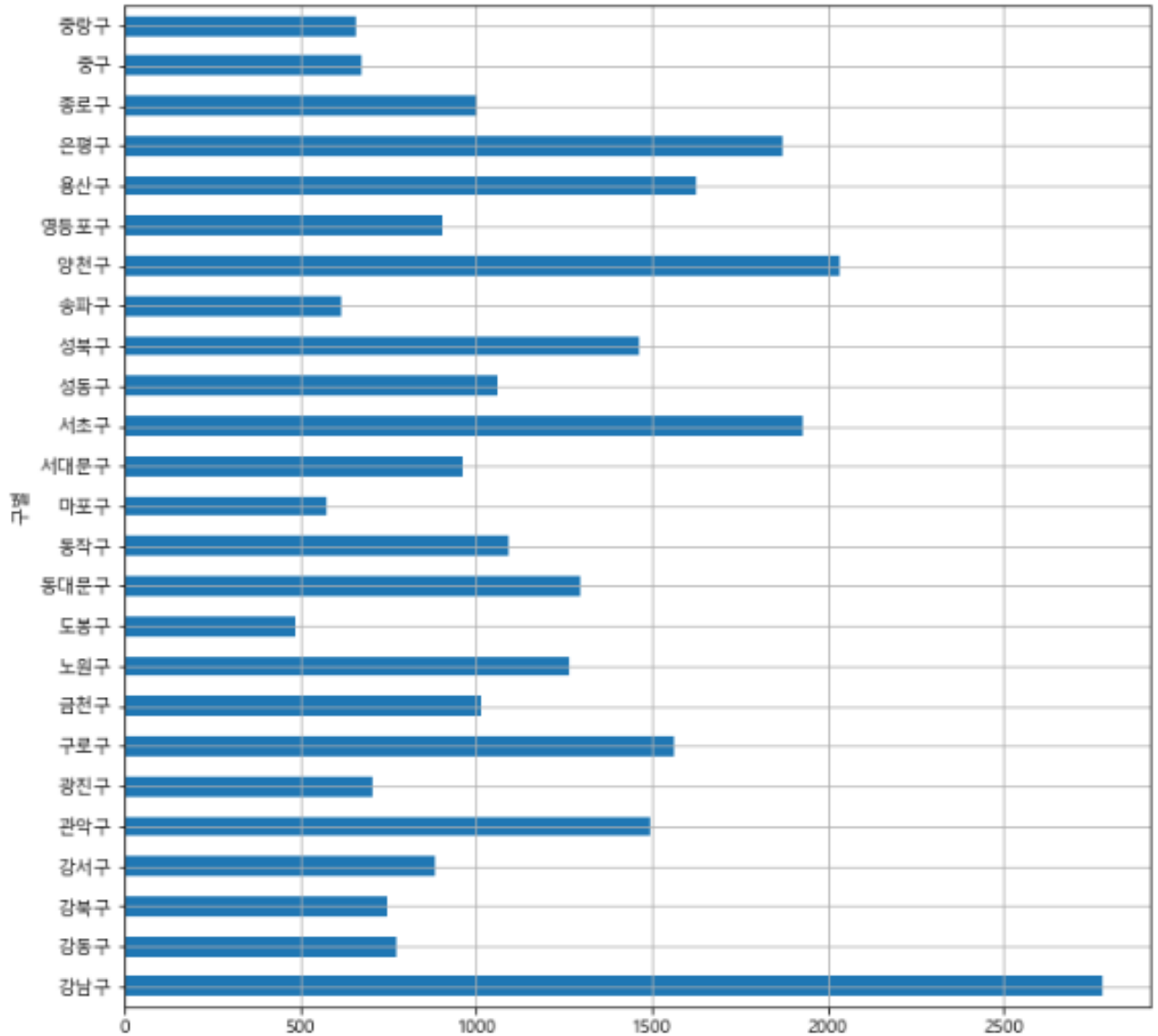
In [126]:
```
import platform

from matplotlib import font_manager, rc
mhplt.rcParams['axes.unicode_minus'] = False

if platform.system() == 'Darwin':
    rc('font', family='AppleGothic')
elif platform.system() == 'Windows':
    path = "c:/Windows/Fonts/malgun.ttf"
    font_name = font_manager.FontProperties(fname=path).get_name()
    rc('font', family=font_name)
else:
    print('Unknown system... sorry~~~~')
```

In [127]:
```
data_result.head()
```

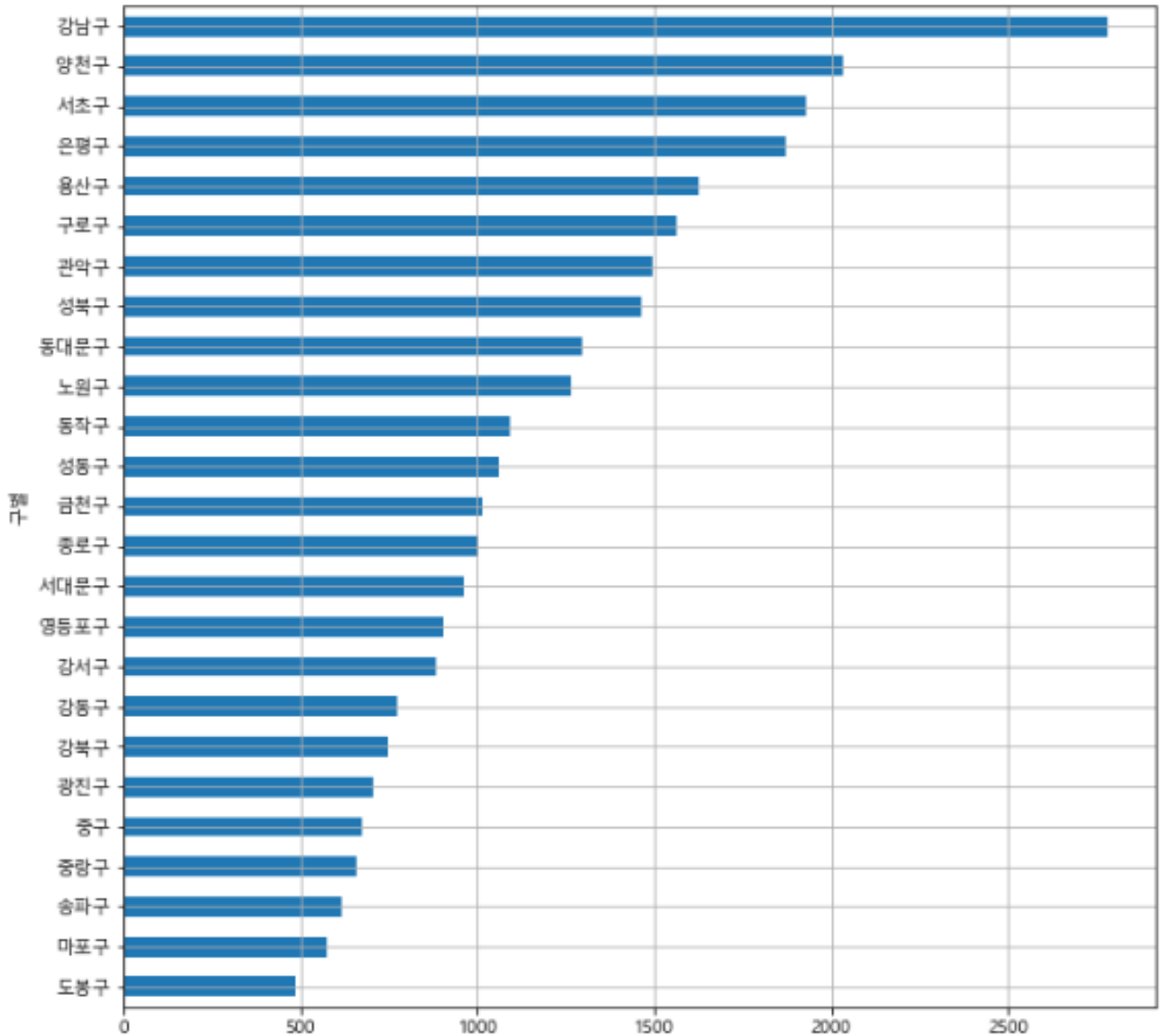
Out[127]:

구별	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율
강남구	2780	150.619195	570500.0	565550.0	4950.0	63167.0	0.867660	11.072217
강동구	773	166.490765	453233.0	449019.0	4214.0	54622.0	0.929765	12.051638
강북구	748	125.203252	330192.0	326686.0	3506.0	54813.0	1.061806	16.600342
강서구	884	134.793814	603772.0	597248.0	6524.0	72548.0	1.080540	12.015794
관악구	1496	149.290780	525515.0	507203.0	18312.0	68082.0	3.484582	12.955291

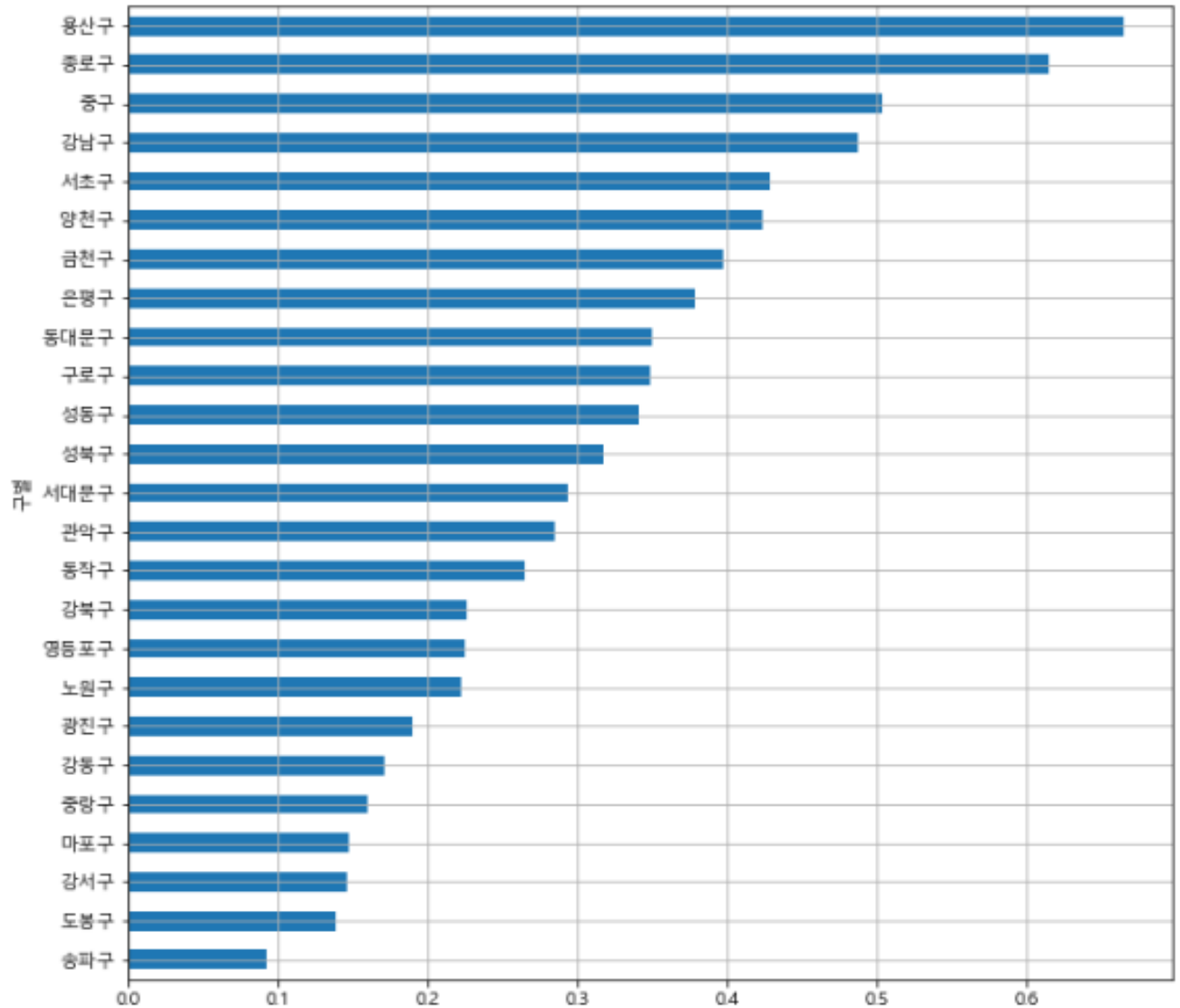
In [128]:
```
mhplt.figure()
data_result['소계'].plot(kind='barh', grid=True, figsize=(10,10))
mhplt.show()
```



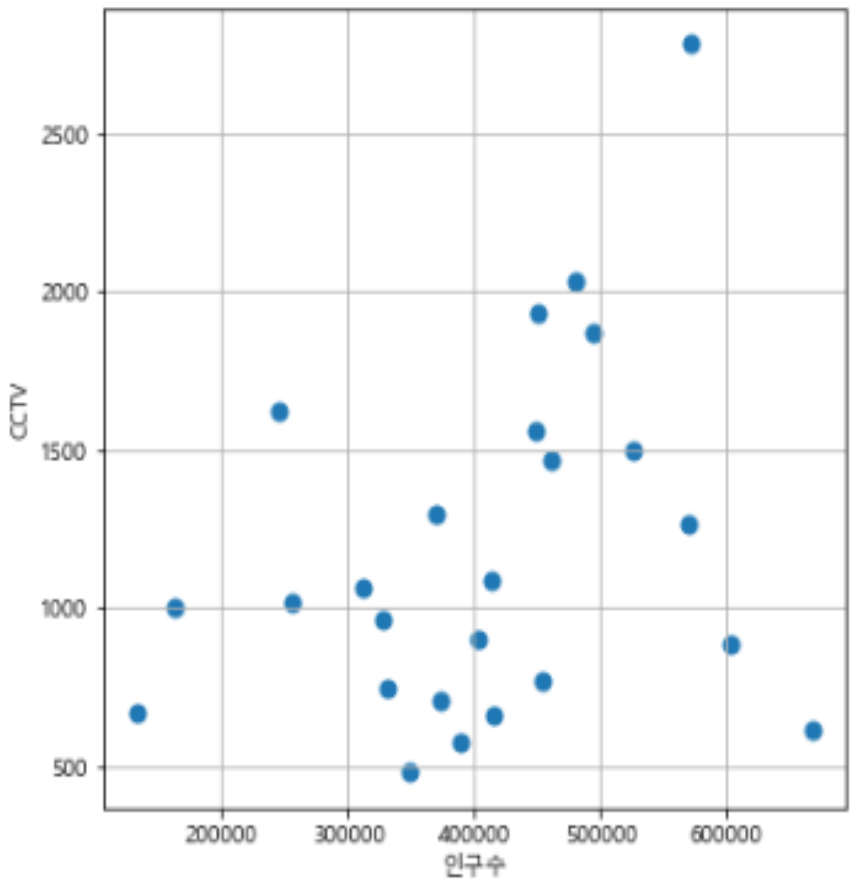
In [129]:
```
data_result['소계'].sort_values().plot(kind='barh',
                                       grid=True, figsize=(10,10))
mhplt.show()
```



In [130]:
```
data_result['CCTV비율'] = data_result['소계'] / data_result['인구수'] * 100

data_result['CCTV비율'].sort_values().plot(kind='barh',
                                           grid=True, figsize=(10,10))
mhplt.show()
```



In [131]:
```
mhplt.figure(figsize=(6,6))
mhplt.scatter(data_result['인구수'], data_result['소계'], s=50)
mhplt.xlabel('인구수')
mhplt.ylabel('CCTV')
mhplt.grid()
mhplt.show()
```

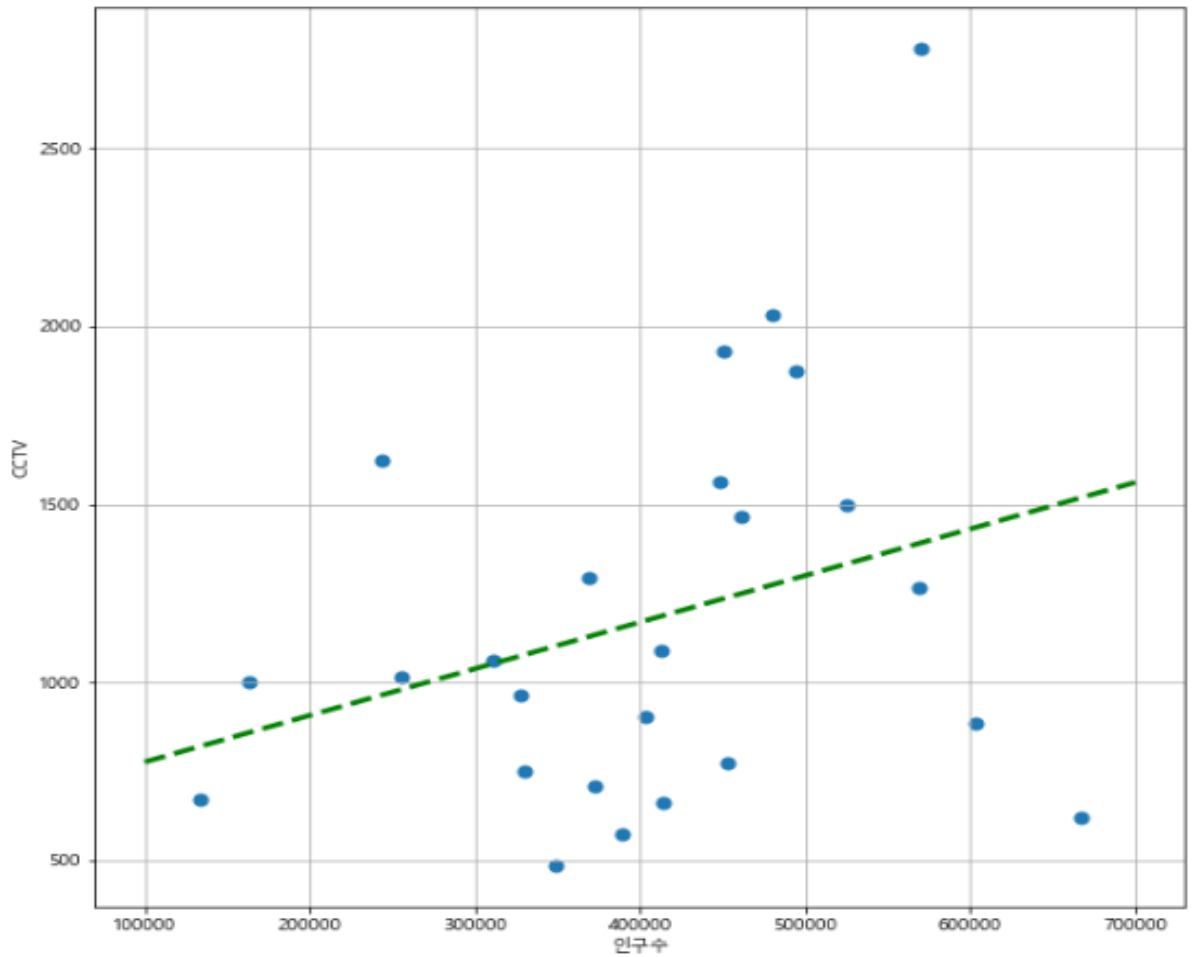


In [132]:
```
fp1 = mhnp.polyfit(data_result['인구수'], data_result['소계'], 1)
fp1
```

Out[132]:
```
array([1.30916415e-03, 6.45066497e+02])
```

In [133]:
```
f1 = mhnp.poly1d(fp1)
fx = mhnp.linspace(100000, 700000, 100)
```

In [134]:
```
mhplt.figure(figsize=(10,10))
mhplt.scatter(data_result['인구수'], data_result['소계'], s=50)
mhplt.plot(fx, f1(fx), ls='dashed', lw=3, color='g')
mhplt.xlabel('인구수')
mhplt.ylabel('CCTV')
mhplt.grid()
mhplt.show()
```

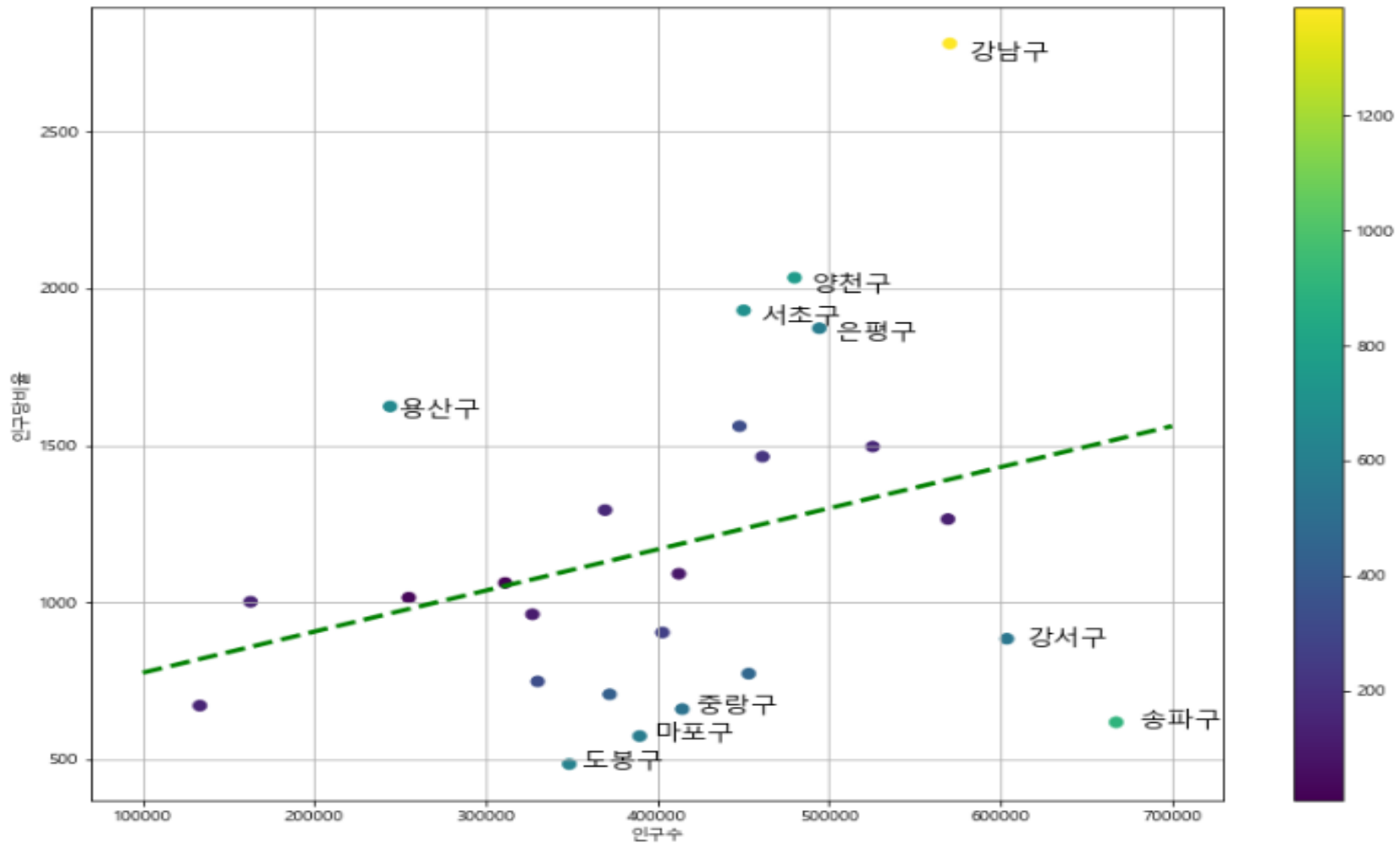


In [135]:
```
fp1 = mhnp.polyfit(data_result['인구수'], data_result['소계'], 1)

f1 = mhnp.poly1d(fp1)
fx = mhnp.linspace(100000, 700000, 100)

data_result['오차'] = mhnp.abs(data_result['소계'] - f1(data_result['인구수']))

df_sort = data_result.sort_values(by='오차', ascending=False)
df_sort.head()
```

Out[135]:

구별	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율	CCTV비율	오차
강남구	2780	150.619195	570500.0	565550.0	4950.0	63167.0	0.867660	11.072217	0.487292	1388.055355
송파구	618	104.347826	667483.0	660584.0	6899.0	72506.0	1.033584	10.862599	0.092587	900.911312
양천구	2034	34.671731	479978.0	475949.0	4029.0	52975.0	0.839413	11.036964	0.423769	760.563512
서초구	1930	63.371266	450310.0	445994.0	4316.0	51733.0	0.958451	11.488308	0.428594	695.403794
용산구	1624	53.216374	244203.0	229456.0	14747.0	36231.0	6.038828	14.836427	0.665020	659.231690

In [136]:
```
mhplt.figure(figsize=(14,10))
mhplt.scatter(data_result['인구수'], data_result['소계'],
              c=data_result['오차'], s=50)
mhplt.plot(fx, f1(fx), ls='dashed', lw=3, color='g')

for n in range(10):
    mhplt.text(df_sort['인구수'][n]*1.02, df_sort['소계'][n]*0.98,
               df_sort.index[n], fontsize=15)

mhplt.xlabel('인구수')
mhplt.ylabel('인구당비율')
mhplt.colorbar()
mhplt.grid()
mhplt.show()
```



7. 소감

첫 과제를 끝내며 느낀 점은 무엇보다 시간이 촉박했던 점입니다.

생각보다 타이핑할게 너무 많아서 하나하나 타이핑하다가
양이 많아서 그래프쪽부터는 복사해서 결과만 봤습니다.

아쉬운점은 주석도 달려고 공간 만들어뒀는데 사진 올려놓고 보니까
시간이 너무 늦어서 달지 못하고 그대로 제출했습니다.

과제 제출이 끝나도 따로 주말쯤엔 1장 다시 공부하려고 합니다.

파이썬 언어는 한번도 배워보지 않은 언어라서 첫날에 걱정하긴 했는데
과제를 할 때 책을 보고 해서 그런건지 몰라도 어느정도 이해가 안되는건 없었지만,
한번도 공부하지 않은 언어로 프로젝트를 하려면 파이썬에 대해서 좀 더 공부해야 할 것 같습니다.